6. What is the difference between executing multiple commands with && and ; in Bash?

The **difference between using `&&` and `;` in Bash** lies in how these operators handle the execution flow based on the success or failure of commands:

---

1. The `&&` Operator

- **Conditional Execution**: Executes the second command **only if** the first command is successful (i.e., the first command exits with a status of `0`).
- **Usage**:

  ```
  command1 && command2
  ```

- **Behavior**:
  - If `command1` succeeds, then `command2` is executed.
  - If `command1` fails (non-zero exit status), `command2` is skipped.
- **Example**:

  ```
  mkdir new_folder && echo "Directory created"
  ```

  - If `mkdir new_folder` is successful, it will print:

    ```
    Directory created
    ```

  - If `mkdir new_folder` fails (e.g., the directory already exists), `echo` will not run.

---

2. The `;` Operator

- **Unconditional Execution**: Executes both commands **regardless of** the success or failure of the first command.
- **Usage**:

  ```
  command1; command2
  ```

- **Behavior**:
  - `command1` runs.
  - Once `command1` finishes (whether it succeeds or fails), `command2` runs.
- **Example**:

  ```
  mkdir new_folder; echo "Attempted to create directory"
  ```

   - Even if `mkdir new_folder` fails, the message:

     ```
     Attempted to create directory
     ```

     will always print.

Key Differences

Feature	`&&` (AND)	`;` (Separator)
Execution Condition	Executes the second command **only if the first succeeds**	Executes the second command **always**, regardless of the first command's result.
Control Logic	Conditional (dependent on success)	Unconditional (independent of success)
Use Case	For commands that depend on the success of the previous one.	For commands that should always run, even if others fail.

---

3. Combining Both

- You can combine `&&` and `;` for more control.

  ```
  command1 && command2; command3
  ```

  - Here:
    - `command2` executes only if `command1` succeeds.
    - `command3` executes regardless of the results of the first two commands.

- **Example**:

  ```
  mkdir new_folder && echo "Success" || echo "Failed"; echo "Done"
  ```

  - If the directory is created:

    ```
    Success
    Done
    ```

  - If the directory creation fails:

    ```
    Failed
    Done
    ```

When to Use Each

- **Use `&&`**:
  - For dependent tasks, such as:

    ```
    compile_program && run_tests
    ```

    - If the program fails to compile, the tests won't run.
- **Use `;`:**
    - For independent tasks, such as:

      ```
      echo "Starting backup"; tar -czf backup.tar.gz /important_data
      ```

    - The `echo` runs regardless of whether the backup succeeds.
